UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

Herramienta de apoyo a la docencia en Sistemas Operativos

PROYECTO FIN DE CARRERA

TEMA: Sistemas Operativos

TÍTULO: Herramienta de apoyo a la docencia en Sistemas Operativos

AUTOR: Erik Wedin

TUTOR: Javier Martín Rueda **VºBº.**

DEPARTAMENTO:

DTE (Departamento de Ingeniería Telemática y Electrónica)

Miembros del tribunal Calificador:

PRESIDENTE: Manuel César Rodríguez Lacruz

VOCAL: Javier Martín Rueda

VOCAL SECRETARIO: Javier Malagón Hernández

Fecha de lectura:

Calificación: El Secretario

RESUMEN DEL PROYECTO

El proyecto fin de carrera es una aplicación Java que pretende ser un apoyo a la docencia en Sistemas Operativos y al entendimiento de la materia por el alumno. Trata de explicar gráficamente y por medio de pruebas el tema de planificación de procesos de un sistema operativo multi-proceso. Facilita al alumno/usuario una interfaz en la que pueda definir su propio algoritmo de planificación a corto plazo y probar distintos ya definidos.

El proyecto fin de carrera de herramienta de apoyo a la docencia en Sistemas Operativos quiere ayudar al alumno a entender el funcionamiento de un planificador a corto plazo. Lo hace mediante una representación gráfica de procesos que ocupan o el procesador o distintas unidades de entrada/salida mientras transcurre el tiempo. El tiempo está dividido en ciclos de reloj de un procesador, a lo que a continuación se referirá como unidades de tiempo. Los procesos están definidos por su nombre, la instante de entrada que entran al sistema, su prioridad y la secuencia de unidades de tiempo en el procesador y unidades de entrada/salida que necesitan para terminar su trabajo.

El alumno puede configurar el sistema a su gusto en cuanto al número y comportamiento de las unidades de entrada/salida. Puede definir que una unidad solo permita acceso exclusivo a los procesos, es decir que solo un proceso puede ocuparla simultáneamente, o que permita el acceso múltiple a sus recursos.

El alumno puede construir un planificador a corto plazo propio, integrarlo en el sistema y ver cómo se comporta. Se debe usar la interfaz Java proporcionada para su construcción.

La aplicación muestra datos estadísticos como por ejemplo la eficiencia del sistema (el tiempo activo de la CPU dividido por el tiempo total de la simulación), tiempos de espera de los procesos, etc. Se calcula después de cada unidad de tiempo para que el alumno pueda ver el momento exacto donde la simulación tomó un giro inesperado.

La aplicación está compuesta por un motor de simulación que contiene toda la lógica y un conjunto de clases que forman la interfaz gráfica que se presenta al usuario. Estos dos componentes pueden ser reemplazados siempre y cuando se mantenga la definición de sus conectores igual.

La aplicación la he hecho de manejo muy simple e interfaz fácil de comprender para que el alumno pueda dedicar todo su tiempo a probar distintas configuraciones y situaciones y así entender mejor la asignatura.

The project is called “Tool to Support Teaching of the Subject Operating Systems” and is an application that aims to help students understand on a deeper level the inner workings of how an operating system handles multiple processes in need of CPU time by the means of a short-term planning algorithm. It does so with a graphical representation of the processes that occupy the CPU and different input/output devices as time passes by. Time is divided in CPU cycles, from now on referred to as time units. The processes are defined by their name, the moment they enter the system, their priority and the sequence of time units they need to finish their job.

The student can configure the system by changing the number and behavior of the input/output devices. He or she can define whether a device should only allow exclusive access, i.e. only one process can occupy it at any given time, or if it should allow multiple processes to access its resources.

The student can build a planning algorithm of his or her own and easily integrate it into the system to see how it behaves. The provided Java interface and the programming language Java should be used to build it.

The application shows statistical data, e.g. the efficiency of the system (active CPU time divided by total simulation time) and time spent by the processes waiting in queues. The data are calculated after passing each time unit in order for the student to see the exact moment where the simulation took an unexpected turn.

The application is comprised of a simulation motor, which handles all the logic, and a set of classes, which is the graphical user interface. These two parts can be replaced individually if the definition of the connecting interfaces stays the same.

I have made the application to be very easy to use and with an easy to understand user interface so the student can spend all of his or her time trying out different configurations and scenarios in order to understand the subject better.

INTRODUCCIÓN 5

DEFINICIÓN DE REQUISITOS 6

GENERAL 6
SIMULACIÓN 6
PROCESO 6
INTERFAZ DE PLANIFICACIÓN 7
INTERFAZ GRÁFICA 7

TECNOLOGÍAS USADAS 8

DISEÑO 9

DESCRIPCIÓN GENERAL DE LAS CLASES 9
PROCESO 10
SITIOS 12
PLAN 14
COLA 15
GESTORES 17
DISPOSITIVOES 19
SIMULADOR 20
ESTADOSISTEMA 23
ESTADISTICA 24
ESTADISTICAPROCESO 26
DIAGRAMA RELACIONAL DE CLASES 27
DIAGRAMAS DE ESTADO 28
SIMULADOR 28
ESPECIFICACIÓN FICHERO DE TEXTO 29
DISEÑO PROCEDIMENTAL 30
PLAN FIFO 30
SIMULADOR 31

GUÍA DE USUARIO 32

SISTEMA 32

UNIDADES DE E/S 33

PLANIFICADOR 33

PROCESOS 34

CICLOS 34

EDITAR PROCESO 35

GUARDAR Y CARGAR 35

SIMULACIÓN 36

CONCLUSIONES 37

BIBLIOGRAFÍA 39

Introducción

La asignatura Sistemas Operativos enseña cómo los sistemas operativos modernos aprovechan al máximo los recursos de un ordenador. El método usado cuando varios procesos a la vez comparten el procesador se llama planificador a corto plazo. El proyecto fin de carrera se inició para cubrir la necesidad de un programa simulador de un planificador a corto plazo.

El objetivo del programa es apoyar al alumno para que pueda comprender qué es lo que hace un planificador a corto plazo, cómo lo hace y analizar en qué situaciones se debe usar uno frente a otro. A su disposición tiene el alumno un programa con una interfaz gráfica fácil de entender y además la posibilidad de construir un algoritmo propio de planificación y ver cómo afectaría a la ejecución de los procesos en la simulación.

El programa le permite al alumno configurar una serie de procesos en un sistema y sus características, como por ejemplo el instante de entrada al sistema y la prioridad del proceso. Seleccionando un algoritmo de planificador a corto plazo el alumno podrá visualizar el comportamiento del sistema operativo y entender de qué manera el algoritmo afecta a la eficiencia del mismo.

Definición de requisitos

General

1. La aplicación permitirá al usuario crear representaciones de procesos (de aquí en adelante referidas como procesos).
2. La aplicación permitirá simular el funcionamiento de un planificador a corto plazo de un sistema operativo.
3. La aplicación deberá ejecutarse en entorno Java, como mínimo de la versión 1.5.

Simulación

4. Se podrá elegir inicialmente uno entre tres algoritmos de planificación a corto plazo:
    a. FIFO (First In First Out)
    b. Round-Robin
    c. SJF (Shortest Job First)
5. Para cada algoritmo de planificación se podrá definir si se tiene en cuenta la prioridad del proceso.
6. Se permitirá al usuario construir un algoritmo de planificación propio y usarlo en el sistema.
7. Los algoritmos propios deben ser desarrollados en lenguaje de programación Java.
8. Se proporcionará una interfaz Java con la especificación de las funciones mínimas que el usuario debe implementar.
9. El sistema manejará una o varias unidades de entrada y salida (E/S).
10. La cantidad de unidades de E/S dependerá del usuario.
11. El usuario podrá definir si la unidad de E/S permitirá el acceso simultáneo.
12. Los tiempos se medirán en unidades de tiempo.
13. Los valores temporales de los procesos serán un número entero.
14. Se podrá reproducir la simulación de principio a fin.
15. Se podrá adelantar paso a paso la simulación.
16. Se podrá volver hacia atrás en la simulación.
17. Se calcularán datos estadísticos de la simulación, como por ejemplo cuánto tiempo ha estado la CPU activa.

Proceso

18. Se van a poder configurar las siguientes características de un proceso:
    a. Nombre del proceso (alfanumérico)
    b. Instante de entrada al sistema
    c. Orden y duración de ráfagas de CPU y E/S
    d. Prioridad
19. Se permitirá guardar la configuración de un proceso en un fichero de texto.

20. Dichos ficheros podrán ser interpretados por la aplicación y convertidos en procesos.
21. La sintaxis del fichero de texto será legible para ser fácilmente interpretado y editado por programas externos, como por ejemplo editores de texto.
22. En un fichero de texto se podrá guardar un proceso individualmente o un conjunto de procesos.

Interfaz de planificación

23. La interfaz de planificación será basada en eventos.
24. La implementación de la interfaz debe manejar la cola o las colas de preparado.

Interfaz gráfica

25. La interfaz gráfica permitirá definir procesos nuevos
26. La interfaz gráfica permitirá definir los parámetros de un proceso.
27. La interfaz gráfica permitirá editar procesos existentes.
28. La interfaz gráfica permitirá guardar un conjunto de procesos en disco para su uso posterior.
29. La interfaz gráfica permitirá cargar un conjunto de procesos de disco para usar en la simulación.
30. La interfaz gráfica permitirá definir ráfagas de CPU o E/S de forma algorítmica:
    a. Se podrá definir que una secuencia de ráfagas de CPU y E/S se repita n veces.
31. La interfaz gráfica permitirá al usuario ir hacia adelante y atrás en la simulación.
32. La interfaz gráfica permitirá al usuario ir hasta un momento especificado de la simulación.
33. La interfaz gráfica mostrará la siguiente estadística:
    a. Tiempo de inactividad de la CPU
    b. Porcentaje mostrando cuánto ha tenido que trabajar la CPU
    c. Tiempo que ha estado un proceso bloqueado

Tecnologías usadas

Para la realización de este proyecto fin de carrera he usado:

- Java SDK 1.7
- NetBeans IDE 8.0
- Java 7 API

Se eligió usar Java por la fácil distribución del programa a básicamente cualquier ordenador corriendo cualquier sistema operativo. La versión 7 es la última de Java en el momento de realizar el proyecto, pero el programa funcionará igual con la versión 5.

NetBeans es un entorno integrado de desarrollo que permite desarrollar programas Java (entre otros tipos) con facilidad. Entre las características más útiles se encuentra la opción de completar el código mientras escribes, dando sugerencias de lo que se quiere poner, y la facilidad con la que se puede navegar por el código y encontrar rápidamente lo que se busca, como por ejemplo dónde se encuentra la definición o la cabecera de un método.

La API online de Java la consulté mucho durante todo el proyecto. Una de las ventajas de usar el lenguaje de programación de Java es la cantidad y calidad de la documentación disponible online.

Diseño

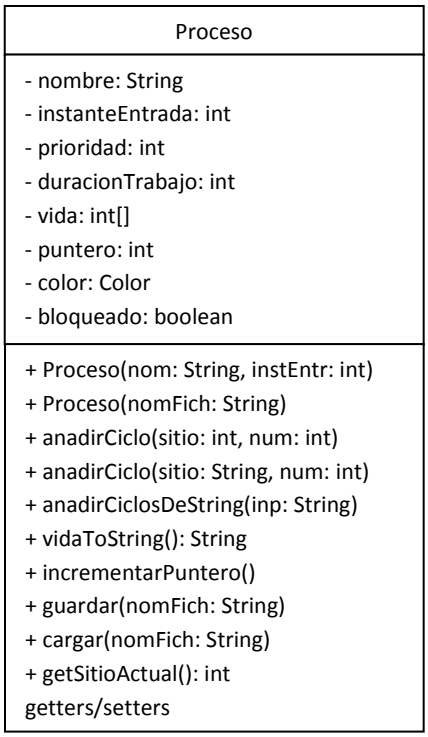
Descripción general de las clases

Clase	Descripción
Proceso	Guarda la información de un proceso, en concreto su nombre, el instante en el que debe entrar al sistema, en qué momentos tiene que pasar de la CPU a E/S y viceversa y la prioridad que tiene.
Sitios	Clase que guarda la relación entre nombre de sitio y un valor entero constante. Sirve para flexibilizar la configuración del sistema (por ejemplo el usuario podrá añadir un número prácticamente infinito de dispositivos).
Cola	Lista ordenada de Procesos. Contiene un número indeterminado de Procesos. Permite ordenarse según cualquier campo de los Procesos mediante un Comparator.
Plan	Interfaz para describir los eventos mínimos que debe tener su implementación.
GestorES	Gestiona los posibles dispositivos de entrada/salida. Se ocupa de meter Procesos en sus colas correspondientes y de gestionar el tiempo que los Procesos ocupan dichos dispositivos.
DispositivoES	Representa una unidad de entrada y salida. Tiene dos modos; multi-acceso o acceso exclusivo. Si es exclusivo, tendrá una Cola asociada.
Simulador	El motor de la aplicación. Se ocupará de proporcionar nuevos procesos a la clase Plan en cuanto entran al sistema, además de manejar el acceso a las unidades de E/S a través de la clase GestorES.
EstadoSistema	Ayuda a la clase Simulador a guardar toda la información relevante al estado del sistema de cada instante de tiempo.
Estadística	Calcula, guarda y devuelve datos estadísticos sobre la ejecución del sistema. Esos datos incluyen por ejemplo el tiempo que cada proceso ha ocupado los distintos dispositivos (CPU, E/S, etc.) y la eficiencia del sistema.
EstadísticaProceso	Sirve para guardar datos de un determinado Proceso.
Fifo	Implementación básica de la interfaz Plan. Es un planificador a corto plazo que asigna a un Proceso tiempo en la CPU según cuando haya entrado al sistema.
ShortestJobFirst	Implementación de la interfaz Plan. Es un planificador que da prioridad a los Procesos cuyo trabajo pendiente en la CPU es el más corto.
Round-Robin	Implementación de la interfaz Plan. Es un planificador que te permite definir un valor cuanto que determina el número de ciclos máximo que puede estar un Proceso ocupando la CPU.

Proceso

Hay cuatro constructores. Puedes crear un proceso sin parámetros, a partir de un nombre y una instante de entrada, a partir de datos procedentes de un fichero de texto o a partir de otra instancia de Proceso (se haría una copia).

Proceso implementa Comparable para que se pueda ordenar según cualquier campo del mismo.



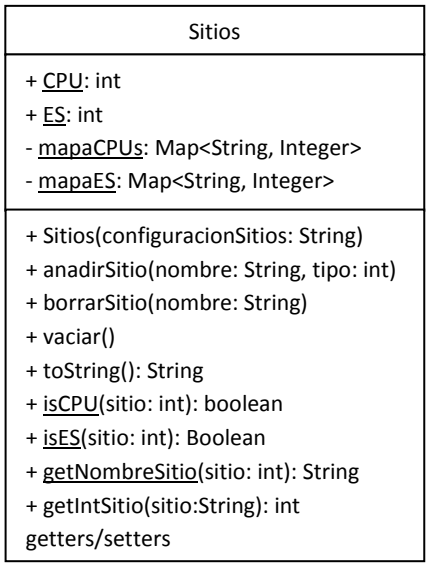
Atributo	Descripción
Nombre	Guarda el nombre del proceso
instanteEntrada	Guarda en qué momento entra el proceso al sistema
prioridad	Guarda el valor de prioridad que puede tener el proceso
duracionTrabajo	Atributo que cambia durante la ejecución de la simulación. Guarda la duración del trabajo pendiente en la CPU.
Vida	Guarda para cada momento de su vida dónde tiene que estar el Proceso, bien sea la CPU o algún dispositivo de entrada/salida
Puntero	Guarda la posición de su vida en la que se encuentra el Proceso en un momento dado
Color	Guarda el color que se le han asignado (para su representación gráfica).
bloqueado	True si el Proceso en ese momento se encuentra en una Cola, false si no.

Método	Descripción
anadirCiclo - sitio - num	Añade un ciclo a la vida del Proceso - Parámetro de entrada. Define el tipo de ciclo que es (dónde necesita estar el Proceso). Puede ser un String o un número entero. - Parámetro de entrada. El número de ciclos a añadir
vidaToString	Devuelve una cadena de caracteres describiendo la vida del Proceso.
guardar - nomFich	Guarda la información del Proceso a un fichero de texto. - Parámetro de entrada. La ruta al fichero resultante.
cargar - nomFich	Carga la información del Proceso desde un fichero de texto - Parámetro de entrada. La ruta del fichero.
datosToString	Devuelve los datos básicos del Proceso en formato String.
incrementarPuntero	Incrementa en una unidad de tiempo el puntero del Proceso. Si el Proceso ha terminado el puntero se pone a -1.
getSitioActual	Devuelve un valor entero que representa el sitio que ocupa actualmente el Proceso. Lanza una excepción si el Proceso ha terminado su vida.
calcularProximoTrabajo	Calcula y guarda la duración del trabajo pendiente en la CPU del Proceso.

Sitios

Se define un sitio como un dispositivo que puede ocupar un Proceso. Por ejemplo la CPU o un dispositivo E/S.

Al constructor de la clase se le puede pasar un String sacado de por ejemplo un fichero de texto de configuración del sistema y la clase lo interpreta y crea los mapas necesarios para el correcto funcionamiento del sistema.

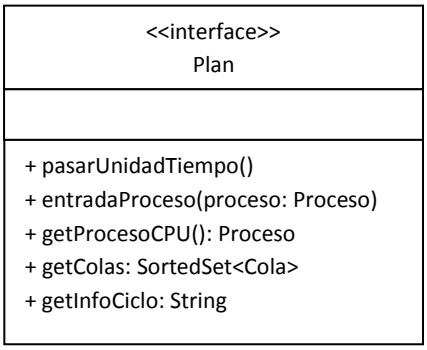


Atributo	Descripción
CPU	Valor entero constante que se usará para definir el tipo del sitio.
ES	Valor entero constante que se usará para definir el tipo del sitio.
mapaCPUs	Variable estática. Mapa que relaciona nombre del sitio del tipo CPU con un valor entero constante.
mapaES	Variable estática. Mapa que relaciona nombre del sitio del tipo ES con un valor entero constante.

Método	Descripción
anadirSitio - nombre - tipo	Añade un sitio al objeto y procura que tenga un valor entero único entre todos los demás sitios. - Parámetro de entrada. Define el nombre del sitio. - Parámetro de entrada. Define el tipo del sitio (entre las constantes de la clase)
borrarSitio - nombre	Elimina el sitio (identificado por nombre) de la lista interna. - Parámetro de entrada. Se buscará el sitio por su nombre, y si es encontrado será borrado de la lista interna.
vaciar	Borra todos los sitios anteriormente guardados. Es como instanciar de nuevo.
isCPU - sitio	Método estático que devuelve true si el sitio es de tipo CPU - Parámetro de entrada. Número entero que corresponde a un sitio.
isES - sitio	Método estático que devuelve true si el sitio es de tipo entrada/salida - Parámetro de entrada. Número entero que corresponde a un sitio.
getNombreSitio - sitio	Método estático que devuelve el nombre de un sitio según el valor entero pasado como parámetro. - Parámetro de entrada. Número entero que corresponde a un sitio.
getIntSitio - sitio	Método estático que devuelve el valor entero asociado a un sitio, identificado por el String pasado como parámetro. - Parámetro de entrada. Un String identificativo de un sitio.
toString	Método sobrescrito que convierte los valores de sus mapas a una línea de texto que puede ser leída e interpretada cuando se carga la configuración del sistema.

Plan

De fábrica vienen tres clases que implementan la interfaz Plan. Simulan el funcionamiento de los algoritmos de planificación a corto plazo FIFO, ShortestJobFirst y Round-Robin.

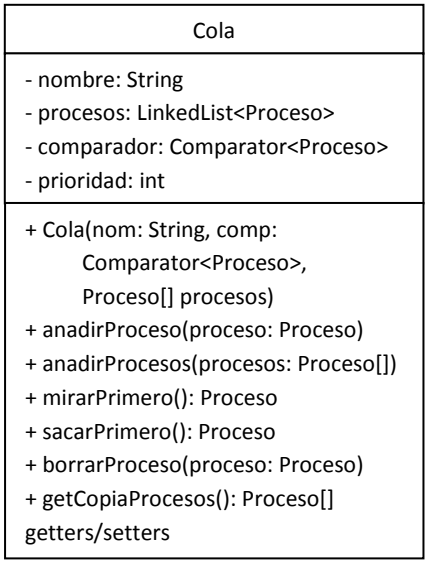


Método	Descripción
pasarUnidadTiempo	Evento llamado por el Simulador cuando ése cree conveniente que el Plan pase una unidad de tiempo.
entradaProceso - proceso	Evento llamado por el Simulador cuando hay un Proceso que necesita trabajar en la CPU. - Parámetro de entrada. El proceso que quiere entrar a la CPU
getProcesoCPU	Devuelve el Proceso que ahora mismo ocupa la CPU.
getColas	Devuelve un conjunto ordenado de las Colas que están en uso.
getInfoCiclo	Devuelve una cadena de caracteres que el programa muestra como información añadida a cada paso de unidad de tiempo.

Cola

Los constructores de la Cola son cualquier combinación de los parámetros nombre, comparador y Procesos. Los valores por defecto son "Cola" para el nombre, un comparador que ordena los Procesos según hayan entrado a la cola y una lista de Procesos vacía.

También existe el constructor que crea una copia de la Cola y todos los Procesos que están en ella.

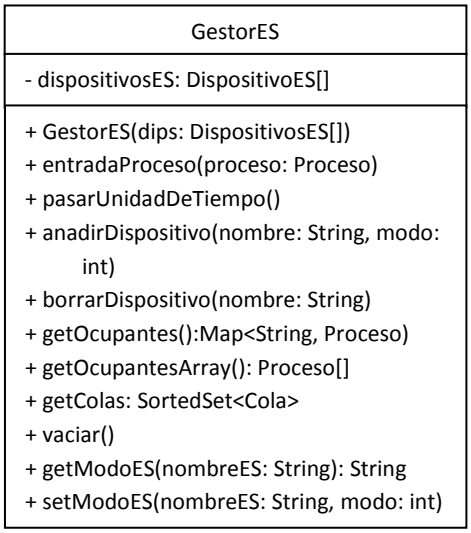


Atributo	Descripción
nombre	Guarda el nombre de la Cola
prioridad	Guarda el valor de prioridad que puede tener la Cola
procesos	Conjunto ordenado que contiene los procesos de la Cola
comparador	Guarda la prioridad que se le han asignado a la Cola

Método	Descripción
anadirProceso - proceso	Añade un Proceso a la Cola. Devuelve true si se ha añadido, false si no se ha podido añadir (porque ya existe uno con el mismo nombre). Si se ha podido meter, cambia el estado del Proceso a bloqueado. - Parámetro de entrada. El Proceso a añadir a la Cola.
anadirProcesos - procesos	Añade un número de Procesos a la Cola. - Parámetro de entrada. El array de Procesos a añadir a la Cola.
mirarPrimero	Devuelve el Proceso en la primera posición de la cola, sin sacarlo
sacarPrimero	Saca y devuelve el Proceso que ocupa la primera posición de la Cola. Cambia el estado del Proceso a no bloqueado.
borrarProceso - proceso	Borra un Proceso de la Cola. - Parámetro de entrada. El Proceso que se quiere borrar.
vacio	Devuelve true si la Cola está vacía, false si no es el caso.
getProcesos	Devuelve un array de los Procesos que contiene la Cola.
getCopiaProcesos	Devuelve un array con una copia de cada uno de los Procesos que contiene la Cola.

GestorES

El constructor con parámetros admite un array de DispositivoES que se quiera añadir al gestor. También existe un constructor sin parámetros.



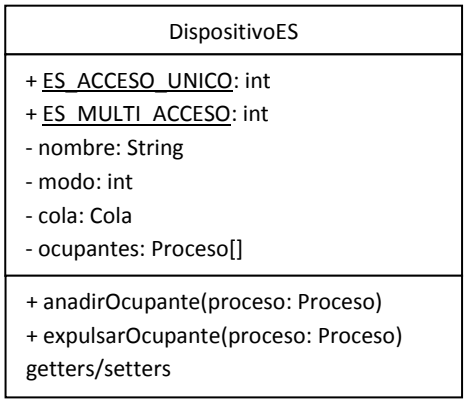
Atributo	Descripción
dispositivosES	Una lista de los dispositivos que el gestor maneja.

Método	Descripción
entradaProceso - proceso	Evento llamado por el simulador en el momento que un Proceso necesita tiempo en cualquier dispositivo de entrada/salida - Parámetro de entrada. El Proceso que quiere tiempo en entrada/salida.
pasarUnidadDeTiempo	Método llamado por el simulador cuando quiere que el gestor haga pasar una unidad de tiempo en sus dispositivos.
anadirDispositivo - nombre - modo	Añade un dispositivo entrada/salida al gestor. - Parámetro de entrada. El nombre del dispositivo a añadir. - Parámetro de entrada. Un número entero que representa el modo del dispositivo (si es acceso único o multi-acceso)
borrarDispositivo - nombre	Borra un dispositivo entrada/salida del gestor. - Parámetro de entrada. El nombre del dispositivo a borrar.
vaciar	Borra todos los dispositivos del gestor.
getOcupantes	Devuelve un mapa con nombre del dispositivo y el correspondiente Proceso que le ocupa.
getOcupantesArray	Devuelve un array con todos los Procesos que están en ese momento ocupando cualquier dispositivo entrada/salida.
getColas	Devuelve un conjunto de las colas de cada uno de los dispositivos
getModoES - nombreES	Devuelve un String identificativo del modo del DispositivoES - Parámetro de entrada. El nombre del dispositivo a consultar.
setModoES - nombreES - modo	Configura el modo del DispositivoES según los parámetros de entrada. - Parámetro de entrada. El nombre del dispositivo a cambiar. - Parámetro de entrada. El modo que se le quiere poner al dispositivo.

DispositivoES

El constructor de DispositivoES permite los parámetros nombre y modo. Si el modo es "acceso único" se crea una cola, si no cola tendrá el valor null. La cola, si existe, estará ordenada según cuando hayan entrado los procesos a ella.

La clase poseerá dos constantes para los diferentes modos de un dispositivo entrada/salida, que son "acceso único" o "multi-acceso".

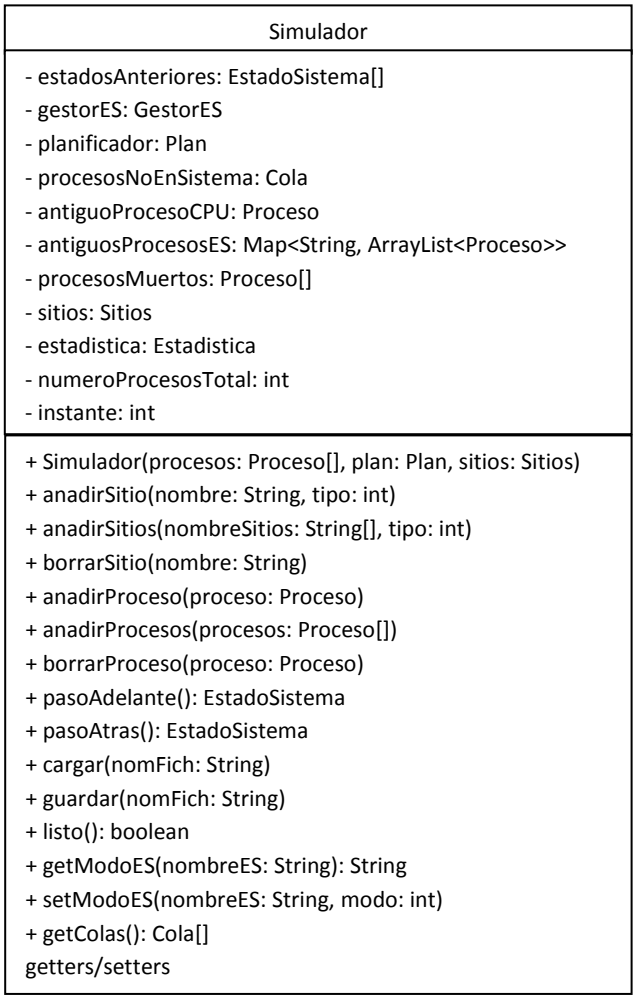


Atributo	Descripción
ES_ACCESO_UNICO	Campo estático que define la constante para el tipo de dispositivo (acceso único)
ES_MULTI_ACCESO	Campo estático que define la constante para el tipo de dispositivo (multi-acceso)
nombre	Un nombre identificativo del dispositivo entrada/salida
modo	Un número entero que puede variar entre las constantes de la clase GestorES.
cola	La cola que le corresponde al dispositivo. Solo se usará si el modo es "acceso único".
ocupantes	Un array de los procesos que actualmente está ocupando el dispositivo.

Método	Descripción
anadirOcupante - proceso	Añade un ocupante al dispositivo. Si ya hay uno y el dispositivo no es de tipo multi-acceso, lanzará una excepción. - Parámetro de entrada. El Proceso que quiere usar el dispositivo de entrada/salida.
expulsarOcupante - proceso	Expulsa un Proceso del dispositivo. - Parámetro de entrada. El Proceso a expulsar.

Simulador

El constructor de esta clase es cualquier combinación de los parámetros procesos, plan y sitios. También existe el constructor sin parámetros. Sin embargo antes de que arranque la simulación es necesario tener todos esos atributos definidos.



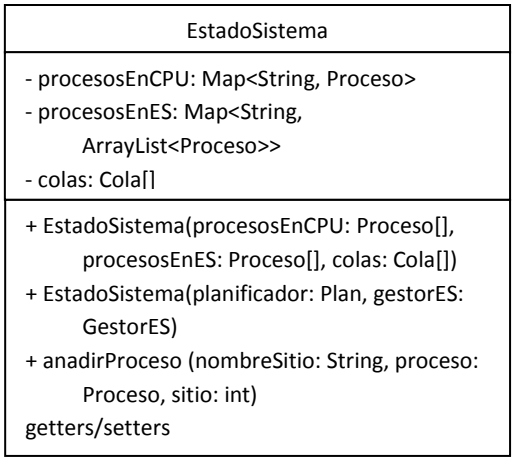
Atributo	Descripción
estadosAnteriores	Array que contiene los estados del sistema, para poder retroceder.
gestorES	El gestor de dispositivos de entrada/salida. Permite llevar el control de los distintos dispositivos de E/S.
planificador	Contiene el planificador que se va a usar en el sistema.
procesosNoEnSistema	Una cola que contiene los procesos que aún no han entrado al sistema.
antiguoProcesoCPU	Guarda el proceso que ocupaba la CPU en el ciclo anterior al actual.
antiguosProcesosES	Mapa que guarda los procesos que ocupaban los distintos dispositivos E/S en el ciclo anterior al actual. El nombre del dispositivo es la clave y el array de Procesos el valor.
procesosMuertos	Un array de los Procesos que ya han terminado su ejecución.
sitios	Contiene valores constantes de los diferentes sitios donde puede estar un proceso en su vida.
estadistica	Contiene el objeto Estadistica para consultar los datos estadísticos de la simulación.
numeroProcesosTotal	Contiene la cantidad de procesos que están en el sistema. Sirve para poder saber si el sistema ha acabado o no.
instante	Número entero que representa el tiempo transcurrido desde el inicio de la simulación.

Método	Descripción
anadirDispositivoES - nombre - tipo	Añade un dispositivo E/S al sistema. - Parámetro de entrada. El nombre del dispositivo E/S que se quiere añadir. - Parámetro de entrada. El tipo del dispositivo E/S. Puede ser acceso único o multi-acceso.
anadirDispositivosES - nombreSitios - tipo	Añade varios dispositivos E/S del mismo tipo al sistema. - Parámetro de entrada. Los nombres de los dispositivos E/S que se quiere añadir. - Parámetro de entrada. El tipo de los dispositivos E/S. Puede ser acceso único o multi-acceso.
borrarDispositivoES - nombre	Borra un dispositivo E/S del sistema. - Parámetro de entrada. El nombre del dispositivo E/S a borrar.
anadirProceso - proceso	Añade un Proceso al sistema. - Parámetro de entrada. El Proceso a añadir al sistema.
anadirProcesos - procesos	Añade varios Procesos al sistema. - Parámetro de entrada. Un array de los Procesos que se quiere añadir.
borrarProceso - proceso	Borra un Proceso del sistema. - Parámetro de entrada. El Proceso a borrar.
pasoAdelante	Sirve para decirle al simulador que pase una unidad de tiempo. Devuelve un objeto EstadoSistema que describe el estado resultante, o null si la simulación ha acabado.
pasoAtras	Sirve para decirle al Simulador que vuelva hacia atrás una unidad de tiempo. Devuelve el objeto del tipo EstadoSistema que describe el estado anterior al actual, o null si no hay estados anteriores.
cargar - nomFich	Configura la simulación desde un fichero de texto. - Parámetro de entrada. El nombre del fichero que contiene la configuración.
guardar - nomFich	Guarda la configuración de la simulación a un fichero de texto. - Parámetro de entrada. El nombre del fichero en el que se quiere guardar la configuración.
listo	Devuelve si la simulación ha terminado o no.
getModoES - nombreES	Devuelve un String identificativo del modo del DispositivoES - Parámetro de entrada. El nombre del DispositivoES que se quiere consultar.
setModoES - nombreES - modo	Configura el modo de un determinado DispositivoES - Parámetro de entrada. El nombre del DispositivoES que se quiere configurar. - Parámetro de entrada. El modo que se le quiere poner al DispositivoES.
getColas	Devuelve un array de todas las Colas que hay en el sistema.

EstadoSistema

El constructor puede ser cualquier combinación de los parámetros procesosEnCPU, procesosEnES y colas, además de ninguno de ellos. También existe un constructor que configura un estado a partir de un Plan y un GestorES.

El método anadirProceso hace una copia del Proceso pasado como parámetro de entrada. Es necesario para diferenciar los estados y que realmente representen una imagen instantánea de la simulación.

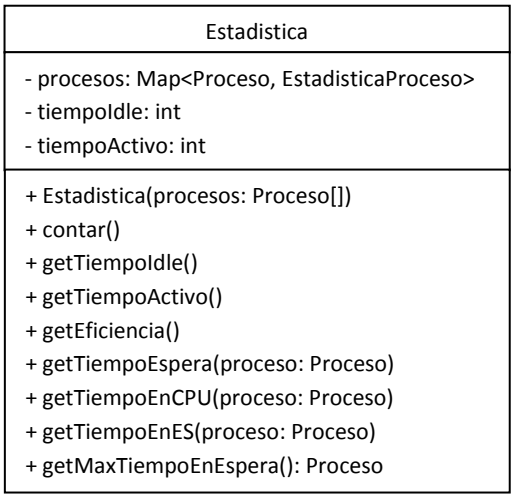


Atributo	Descripción
procesosEnCPU	Mapa de String a Proceso. El String contiene el nombre de la CPU y el Proceso contiene una copia del Proceso que ocupaba esa CPU en ese momento.
procesosEnES	Mapa de String a array de Procesos. El String contiene el nombre de la unidad de entrada/salida y el array de Procesos contiene copias de los Procesos que ocupaban esa unidad de entrada/salida en ese momento.
colas	Array que contiene todas las colas del sistema.

Método	Descripción
anadirProceso	Añade una copia del Proceso al EstadoSistema.
- proceso	- Parámetro de entrada. El Proceso que se quiere añadir.
- sitio	- Parámetro de entrada. El valor constante que coincide con lo que tiene la clase Sitios.
getOcupante	Devuelve el o los Procesos que en este EstadoSistema están ocupando el sitio identificado por el parámetro de entrada. Devuelve null si está vacío.
- sitio	- Parámetro de entrada. Número entero que identifica al sitio que se quiere consultar.

Estadistica

El constructor admite un array de Procesos o sin parámetros con lo que se creará un mapa vacío.



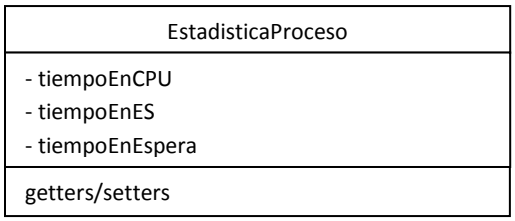
Atributo	Descripción
procesos	Mapa que relaciona cada Proceso en el sistema a un objeto EstadisticaProceso.
tiempoIdle	Guarda el tiempo que ha estado la CPU inactiva.
tiempoActivo	Guarda el tiempo que ha estado la CPU activa.

Método	Descripción
anadirProceso - proceso	Añade un Proceso para empezar a contar sus datos. - Parámetro de entrada. El Proceso a añadir.
anadirProcesos - procesos	Añade varios Procesos para empezar a contar sus datos. - Parámetro de entrada. Los Procesos a añadir.
borrarProceso - proceso	Borra un Proceso de la lista para dejar de contar sus datos. - Parámetro de entrada. El Proceso a borrar.
contar	Método que es llamado por el Simulador cuando pasa una unidad de tiempo en la simulación.
getTiempoIdle	Devuelve el tiempo que ha estado la CPU sin hacer nada.
getTiempoActivo	Devuelve el tiempo que ha estado la CPU trabajando.
getEficiencia	Devuelve el porcentaje del tiempo que ha estado la CPU ocupada.
getTiempoEspera - proceso	Devuelve el tiempo que el Proceso pasado como parámetro ha estado en espera. - Parámetro de entrada. El Proceso del que se quiere obtener su tiempo de espera.
getTiempoEnCPU - proceso	Devuelve el tiempo que el Proceso pasado como parámetro ha estado ocupando la CPU. - Parámetro de entrada. El Proceso del que se quiere obtener su tiempo de ocupación de la CPU.
getTiempoEsperaES - proceso	Devuelve el tiempo que el Proceso pasado como parámetro ha estado ocupando la E/S. - Parámetro de entrada. El Proceso del que se quiere obtener su tiempo de ocupación de la E/S.
getMaxTiempoEnEspera	Devuelve el Proceso que ha estado más tiempo en espera en la simulación.

EstadisticaProceso

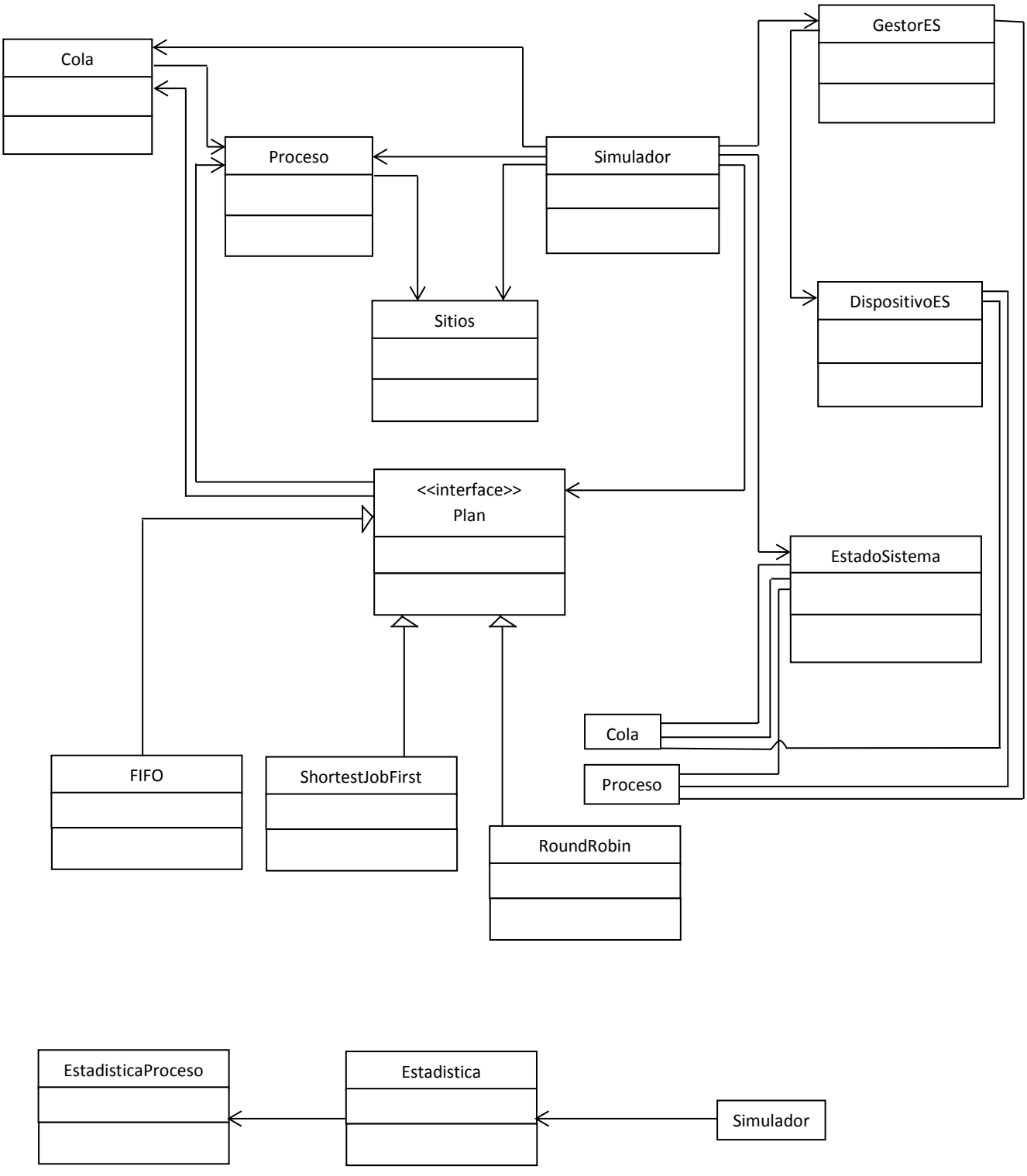
El constructor no tiene parámetros.

La clase sirve para guardar valores temporales asociados a un Proceso.



Atributo	Descripción
tiempoEnCPU	Guarda el tiempo que ha estado el Proceso asociado ocupando la CPU.
tiempoEnES	Guarda el tiempo que ha estado el Proceso asociado ocupando la E/S.
tiempoEnEspera	Guarda el tiempo que ha estado el Proceso asociado en espera.

Diagrama relacional de clases

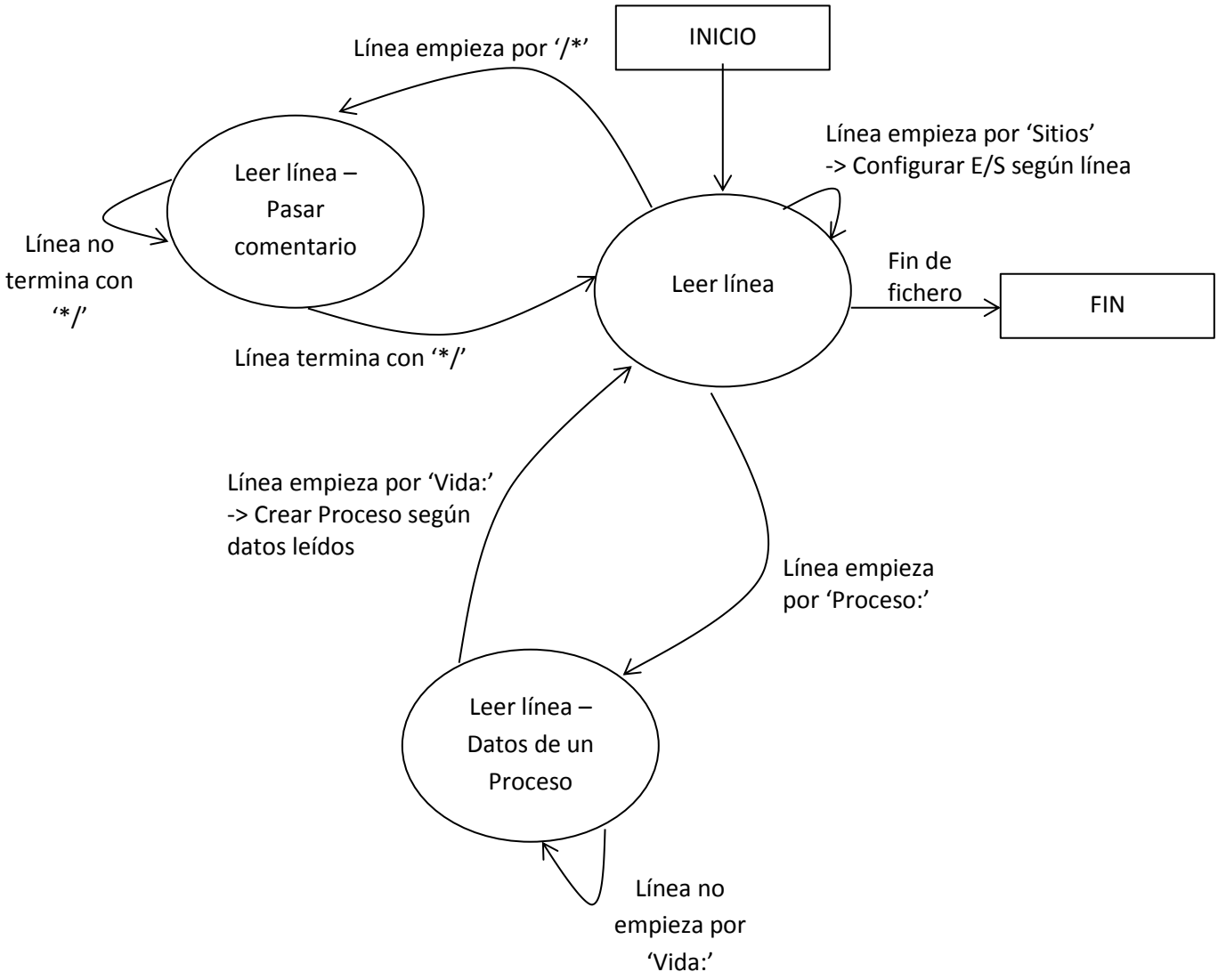


Diagramas de estado

Simulador

Método cargar

El método cargar se encarga de leer un fichero de texto línea por línea y sacar así la configuración del sistema.



Cuando empieza a leer el fichero, busca por unas secuencias de caracteres clave ("Proceso", "Sitios", "Vida", "/*" y "*/") y en cuanto encuentra alguna de ellas, pasa de un estado a otro. Cuando haya llegado al final del fichero, sale del método.

Especificación fichero de texto

El programa guardará a demanda su estado cuando el usuario quiera. También es posible crear un estado del sistema mediante un editor texto cualquiera. Hay que seguir estas pautas.

- Para poner un comentario, es decir, texto que no va a ser interpretado por el programa, se le pone '/*' donde empieza y '*/' donde termina.
- La definición de la CPU y las unidades de entrada/salida (Sitios) es de la siguiente manera:

  Sitios.**tipo**: <**nombre de unidad**>

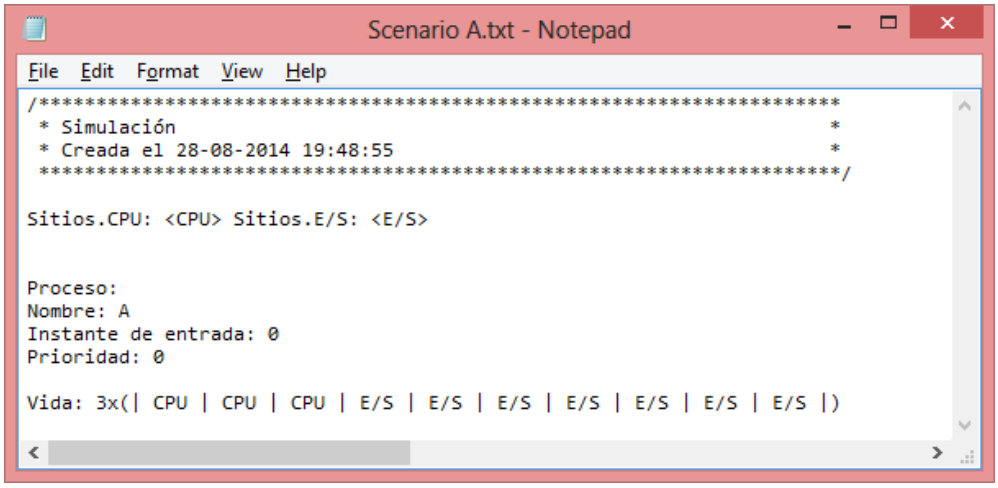
  Por ejemplo "Sitios.CPU: <CPU> Sitios.E/S: <E/S>". Las palabras Sitios.CPU y Sitios.E/S son obligatorias.
- La definición de los Sitios tiene que preceder la de los Procesos.
- Los datos de un Proceso empiezan por la palabra clave "Proceso:" y terminan con la línea que empieza por "Vida:". Entre medias se encuentran los datos del Proceso, o sea su nombre, su instante de entrada y su prioridad. Las palabras en la especificación de la vida del Proceso tiene que coincidir con los nombres de la definición de los Sitios.

  Ejemplo: Proceso:
  Nombre: A
  Instante de entrada: 0
  Prioridad: 0

  Vida: | CPU | CPU | CPU | E/S | E/S | E/S | CPU |

  - En el apartado de vida del Proceso se puede repetir una serie de ciclos de esta manera:

    Vida: 3x(| CPU | CPU | CPU | E/S | E/S | E/S | CPU |)
- La definición del Proceso se repite las veces que se quiera para insertar una multitud de ellos.
- El fichero se guarda con la extensión ".txt".



Diseño procedimental

Solo he incluido aquí los métodos que tienen un cierto grado de complejidad para explicarlos mejor. Las clases Round-Robin y ShortestJobFirst heredan de FIFO. Round-Robin comprueba el cuanto con cada unidad de tiempo para decidir si se expulsará el proceso que ocupa la CPU. ShortestJobFirst ordena la cola de preparados por la duración del trabajo que les queda a los Procesos.

Plan FIFO

Método pasarUnidadTiempo

```
SI está vacío CPU ENTONCES
   sacar primer Proceso de cola preparados
   meter Proceso en CPU
SINO
   incrementar unidad de tiempo para Proceso en CPU
   ver sitio de Proceso
   SI sitio es distinto a CPU ENTONCES
      expulsar Proceso
      sacar primer Proceso de cola preparados
      meter Proceso en CPU
   FINSI
FINSI
```

Simulador

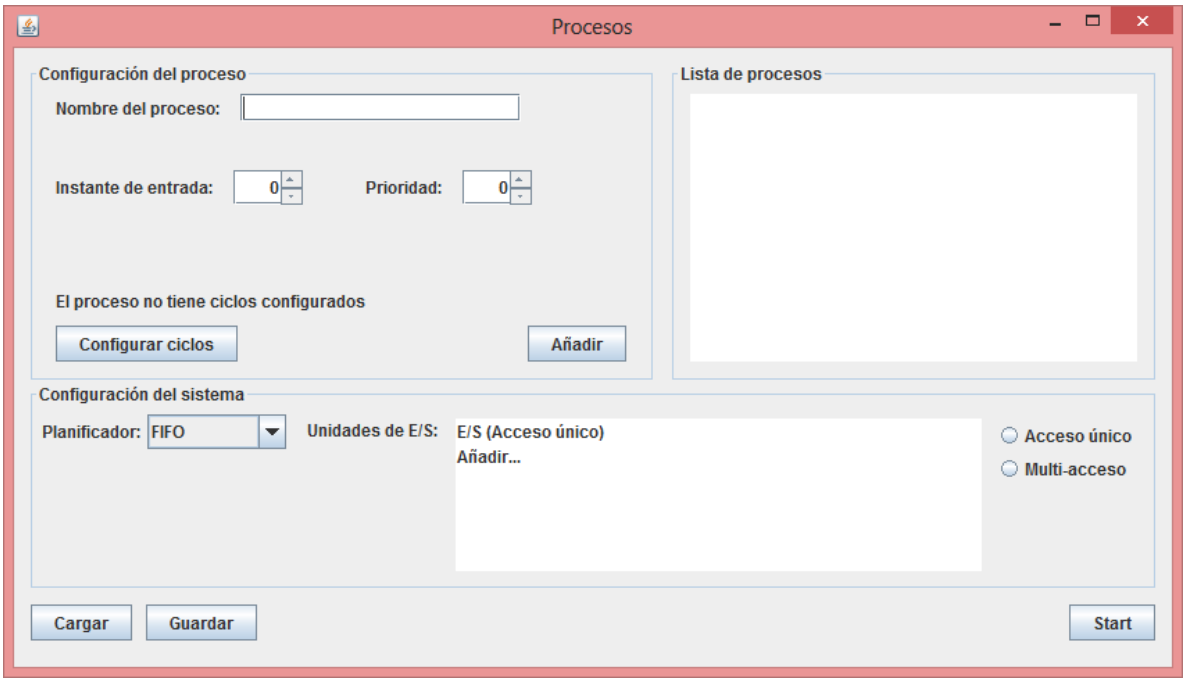
Método pasoAdelante

```
SI instante IGUAL QUE el tamaño de estadosAnteriores ENTONCES
   MIENTRAS el Proceso en la primera posición de la cola "No en sistema" tiene instante
         de entrada IGUAL QUE instante actual HACER
      sacar el Proceso en la primera posición de la cola
      entregarle al Planificador el Proceso sacado
   FIN MIENTRAS

   nuevosProcesosES = Procesos que ocupan los distintos dispositivos E/S
   PARA TODOS dispositivos E/S en nuevosProcesosES HACER
      esProcesos = los procesos que ocupaban el dispositivo E/S en el ciclo anterior de
            la simulación
      SI esProcesos no es null ENTONCES
         PARA TODOS procesos en esProcesos HACER
            SI proceso NO está contenido en nuevoProcesosES ENTONCES
               entregarle al Planificador proceso
            FINSI
         FINPARA
      FINSI
   FINPARA
   antiguosProcesosES = nuevosProcesosES

   nuevoProcesoCPU = Proceso que ocupa la CPU
   SI antiguoProcesoCPU no es null Y nuevoProcesoCPU es distinto a antiguoProcesoCPU
         ENTONCES
      SI antiguoProcesoCPU va a E/S ENTONCES
         entregarle al GestorES antiguoProcesoCPU
      SINO SI antiguoProcesoCPU ha terminado su vida ENTONCES
         AÑADIR antiguoProcesoCPU a procesosMuertos
      FIN SI
   FIN SI
   antiguoProcesoCPU = nuevoProcesoCPU

   estado = nuevo EstadoSistema a partir del planificador y gestor E/S
   contar estadistica

   decirle al Planificador que pase una unidad de tiempo
   decirle al GestorES que pase una unidad de tiempo

   INCREMENTAR instante
   DEVOLVER estado
SINO
   INCREMENTAR instante
   DEVOLVER el estado en la posición instante-1 de estadosAnteriores
FIN SI
```

Guía de usuario

Lo primero con lo que nos encontramos al abrir el programa es la ventana de configuración del simulador. Consiste en tres zonas.

1. Configuración del Proceso donde podemos cambiar o acceder a:
    a. Nombre del Proceso
    b. Nombre corto del Proceso
    c. Instante de entrada
    d. Prioridad
    e. Botón para añadir el Proceso a la simulación
    f. Botón para configurar la vida del Proceso (en qué momento pasa de la CPU a una unidad E/S, por ejemplo)
2. Lista de Procesos. Se muestran los Procesos que se han configurado para el sistema.
3. Configuración del sistema. Se puede elegir el planificador y el número y configuración de las unidades de E/S.

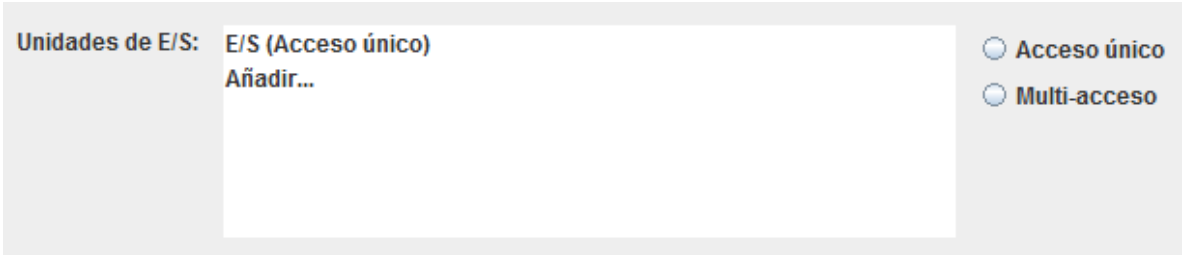


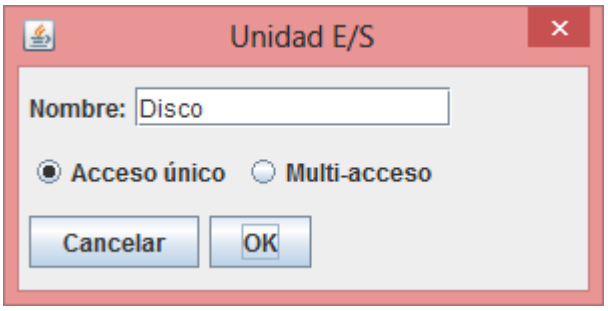
Sistema

En la parte de la configuración del sistema se puede elegir qué planificador se quiere usar en la simulación, datos adicionales del mismo (si procede) y por último el nombre y modo de las unidades de E/S

Unidades de E/S

Antes de definir Procesos se debe establecer el número y modo de los dispositivos E/S que se necesitan. Así se podrá definir la vida del Proceso teniendo en cuenta todos los dispositivos de la simulación. Para añadir una unidad de E/S se pulsa “Añadir…”



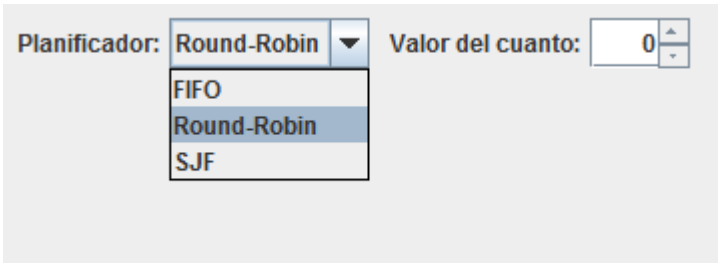


Aparecerá una ventana en la que se puede poner un nombre del dispositivo y elegir si permite o no el acceso simultáneo. Para guardar los datos en el sistema se pulsa “OK”.

Si se quiere borrar una unidad de E/S se selecciona la unidad en la lista y se pulsa la tecla “Suprimir”.

Los botones al lado de la lista son un acceso directo al modo de la unidad de E/S. Para cambiarlo se puede seleccionar la unidad y pulsar uno de los dos botones.

Planificador

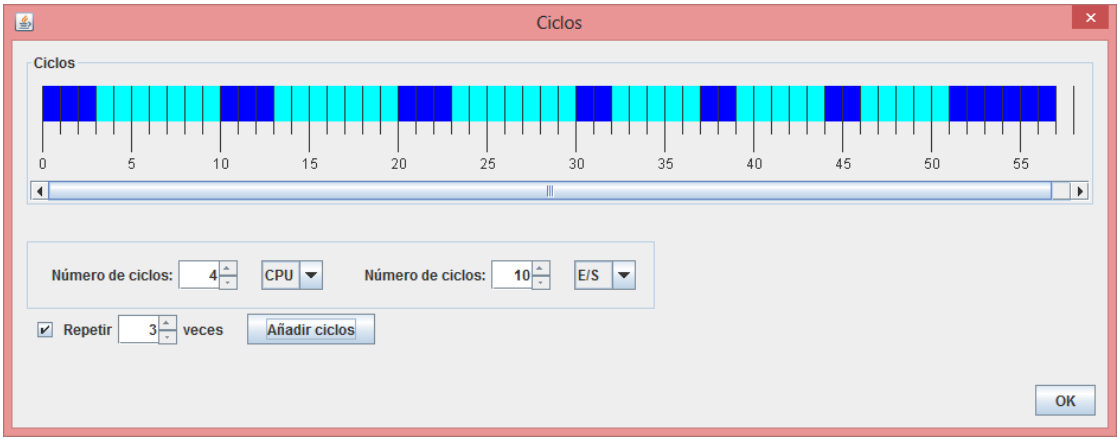


Se puede elegir un planificador pulsando sobre el menú “Planificador”. Si tiene alguna opción adicional, como es en este caso el valor del cuanto, se puede definirlo a continuación.

Procesos

Para definir un nuevo Proceso se debe seguir estos pasos:

1. Definir un nombre para el Proceso. El nombre corto lo sugiere el sistema, pero es editable. Como mucho puede ser de 3 letras.
2. Elegir instante de entrada.
3. Elegir una prioridad para el Proceso.
4. Configurar ciclos.
5. Añadir el Proceso a la simulación pulsando el botón “Añadir”.



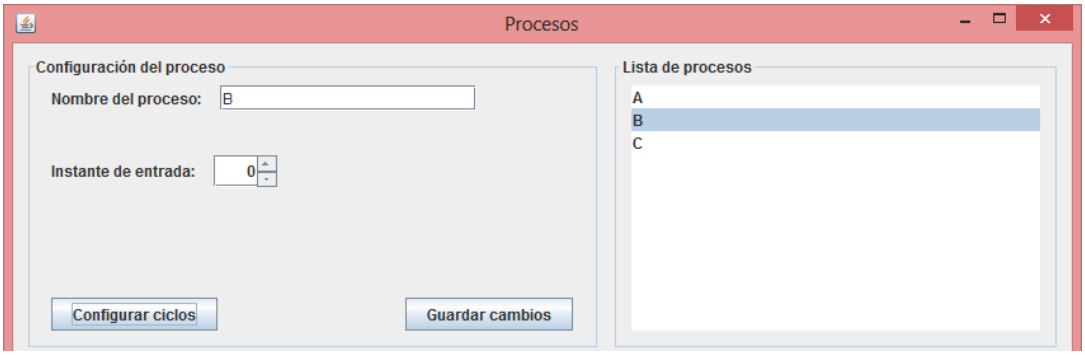
Ciclos

Para definir la vida del Proceso:

1. Poner el número de ciclos y elegir en qué dispositivo los tiene que disfrutar el Proceso.
2. Marcar la casilla “Repetir” y poner cuántas veces (opcional).
3. Pulsar “Añadir ciclos” y se insertan por la derecha.
4. Repetir pasos 1-3 las veces necesarias.
5. Pulsar “OK”.

Editar Proceso

En la lista de Procesos, para borrar uno, se puede marcar el o los Procesos a borrar y pulsar la tecla “Suprimir”. Si se quiere editar un Proceso, se hace doble clic y sus datos se rellenan en la parte de la configuración del Proceso.



Para guardar los cambios realizados se pulsa “Guardar cambios”.

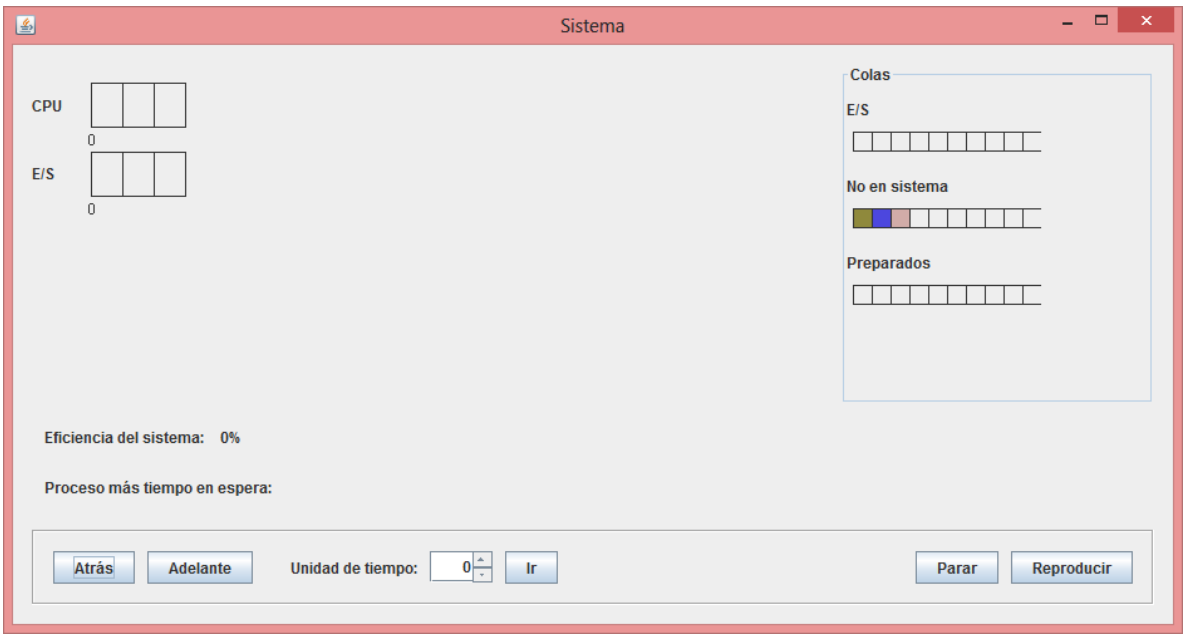
Guardar y cargar

Se ofrece la posibilidad de guardar la configuración actual del sistema mediante el botón “Guardar”. Preguntará por una carpeta destino y un nombre de fichero.

Con el botón “Cargar” se puede abrir un fichero anteriormente guardado.

Simulación

Cuando se haya terminado con la configuración del sistema se pulsa el botón “Start” para empezar la simulación. La ventana que se mostrará es de esta forma.



1. En la parte superior izquierda se ven los dispositivos del sistema. Cada cuadradito representa una unidad de tiempo.
2. En la parte superior derecha se ve el estado de las colas. Cada cuadradito representa una posición dentro de la misma.
3. Debajo de esas dos zonas se encuentran los datos estadísticos de la ejecución de la simulación, como por ejemplo la eficiencia del sistema o el Proceso que más tiempo ha estado en espera.
4. En la parte inferior de la ventana se encuentran los controles de la simulación.

Para adelantar se puede pulsar o el botón “Adelante” o “Reproducir”. Si se pulsa reproducir el programa adelanta la simulación automáticamente paso a paso con un intervalo determinado, hasta pulsar el botón “Parar” o al acabar su vida todos los Procesos. Con el botón “Atrás” se puede ir hacia atrás en la simulación para ver estados anteriores del sistema.

Para ir a un momento de la simulación en concreto se puede introducir el número de la unidad de tiempo en el cuadrado etiquetado “Unidad de tiempo” y pulsar el botón “Ir”.

Conclusiones

La experiencia de crear una aplicación desde cero ha sido muy gratificante. Desde diseñar un programa desde cero que cumpliera los requisitos especificados hasta pensar y realizar lo que el usuario quiere sacar de él. He aprendido mucho sobre cómo funciona un sistema operativo y espero poder transmitir eso al usuario. He creado una interfaz limpia e intuitiva para que el usuario se sienta cómodo con ella desde el principio. Las partes de configuración del sistema y la de los procesos están claramente separadas y el flujo de crear una simulación se entiende y se aprecia. El formato de los ficheros de texto que se usa para guardar la configuración de la simulación está diseñado con el objetivo de máxima legibilidad y facilidad de edición. El usuario puede crear uno nuevo siguiendo unas pocas pautas mediante cualquier editor de texto. Cuando empieza la simulación se muestra al usuario información útil para que pueda analizar la situación con rapidez y eficacia.

La mayor dificultad que he encontrado con el desarrollo del proyecto ha sido la maquetación de los componentes gráficos. Empecé con el editor WYSIWYG (What You See Is What You Get) integrado en NetBeans que te permite componer ventanas arrastrando componentes. No era lo que buscaba, ya que cuando habías acabado con el diseño de una ventana y querías añadir algo nuevo se rompió casi siempre el diseño. Además el código generado era difícil de seguir, por lo que no pude arreglarlo ni en el editor ni en el código fuente. Por eso decidí crearlo enteramente a partir código. No se quitaron los fallos de diseño pero por lo menos tenía más confianza para encontrarlos y arreglarlos.

Otra dificultad con la que he luchado ha sido encontrar la mejor forma de hacer el programa abierto a extensiones. Uno de los requisitos más importantes fue el número 6 (*Se permitirá al usuario construir un algoritmo de planificación propio y usarlo en el sistema*). Empecé por mal camino y construí una interfaz en la que hubo que tener en cuenta todos los aspectos de la simulación, incluso las unidades de entrada/salida. Tuve que separar las funciones porque el usuario solo debería preocuparse de crear un planificador de corto plazo, es decir que solo maneja la o las colas de preparado y qué proceso debe entrar a la CPU en qué momento. Con la ayuda del tutor llegué a una especificación de interfaz ligera y comprensible a la vez que potente. Esas características las comparten las clases Cola y Proceso para cubrir todas las necesidades que pueda tener el usuario.

Aprendí de manera dura la importancia de estructurar el trabajo desde el principio e ir paso a paso de una fase del proyecto a otro. Al principio hice un diseño rápido para empezar a codificar, pensando que eso era suficiente. Perdí bastante tiempo al ver que no era sostenible y tener que volver a empezar. No pasé a la siguiente fase hasta estar totalmente satisfecho con la actual. Eso no quiere decir que en ciertos momentos no tuviera que cambiar algunas cosas del diseño, pero me di cuenta de lo fácil que era cuando tenías un buen diseño del que partir.

Fue un error el no haber quedado más a menudo con el tutor. Por falta de tiempo por temas familiares no tuve mucho tiempo para invertir en el proyecto. No quería reunirme con él sin

haber llegado a un punto en concreto, como por ejemplo poder enseñar el funcionamiento básico del programa. Eso causó que intentase adelantar demasiado temprano la fase de implementación que al final se convirtió en tiempo perdido. Si nos hubiéramos visto más, aunque solo hubiera avanzado lo más mínimo, habría valido la pena. Te puede hacer ver cosas que no habías pensado y apoyarte y tranquilizarte bastante en el desarrollo del proyecto.

Bibliografía

Java Platform, Standard Edition 7 API Specification
Oracle
http://docs.oracle.com/javase/7/docs/api/
Apartados de las clases Map, HashMap, TreeMap, LinkedList, JFrame, JSpinner, JList, DefaultListModel, entre otros

How to Use GridBagLayout (The Java Tutorials)
Oracle
http://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html

How to Use Combo Boxes (The Java Tutorials)
Oracle
http://docs.oracle.com/javase/tutorial/uiswing/components/combobox.html

Quantifiers (The Java Tutorials)
Oracle
http://docs.oracle.com/javase/tutorial/essential/regex/quant.html

Gestión del Procesador
Diapositivas de la asignatura Sistemas Operativos Curso 2009-2010
Departamento de Ingeniería y Arquitectura de Telecomunicaciones
Escuela Universitaria Ingeniería Técnica de Telecomunicaciones